DP-Unified Platform

DP-Unified REST API

Revision 1.7
03/10/2022

Device Partner, LLC

13284 Pond Springs Road, STE 506
Austin, TX 78729-7183

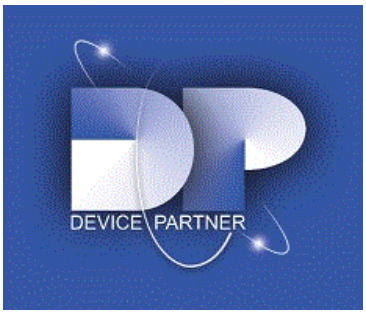

Table of Contents

DP-Unified REST API

1. Introduction

The DP-Unified REST API was designed to give developers the means to access the DP-Unified platform objects and interact with these objects directly without having to go through the DP-Unified GUI (Desktop/Web). This allows the developers more flexibility to directly interact with the objects within the DP-Unified framework to build complex logic within their code (scripts/programming languages) that they would not be able to easily do with the DP-Unified standard GUI approach.

2. Unified Objects

Unified objects represent the workflows of the DP-Unified framework. These objects are basic building blocks that when they are arranged and composed with each other provides the ability to perform complex validation of screens and automate the test case execution on devices. They are designed to be re-usable and easy to create by point and click and make it simple for everyone to use to build their automated test cases. The main object of the DP-Unified framework is to create a test case on one device and run it on all devices.

2.1 Smart Validation Objects

Smart Validation objects are used to validate a screen to make sure they are correct and the that all appropriate images, logo, texts, and or video and audio are present. There are currently 4 types of the Smart Validation objects:

1. ValidateAudio: Responsible for validating if there is audio cutoff on the device in the case that there is valid video being played.
2. ValidateImage: Responsible for validating images/logos that are supposed to be on the screen.
3. ValidateText: Responsible for validating texts on the screen.
4. ValidateVideo: Responsible for validating if video if frozen, black screen or pixelated.

2.2 UISet Objects

Represent user interactions (navigations) on the device are captured as UISet objects. Anytime the user clicks on the buttons on the remote skin the commands are added to the Commands grid. From the commands grid the user can edit/remove commands down the list of commands they want to save to an USet. Once, saved they can select the UISet to play back on one or multiple devices (STBs) at the same time without having to re-enter the commands again. These UISets are then become a building block for the steps in the test case. The UISet can be

commands to navigate the device or a Smart Validation object command to validate the screen of the device.

2.3 UITestCase Objects

Represent Test Cases. Each test case contains 1 to N number of test steps. Where each test step is an UISet.

2.4 UIJob Objects

Represent Jobs. Each job can contain 1 to N number of test cases. This allows you to build complex Job objects that execute related test cases.

2.5 JobRun Objects

Represent JobRuns. Each job run can contain 1 to N number of jobs. This allows you to build complex job run objects that execute a whole test plan.

3. DP-Unified REST API Methods

Each DP-Studio deployed will have its own REST API service module installed on its system. It is recommended the remote system accessing the REST APIs on the DP-Studios to be on the same LAN as the DP-Studio systems.

*Note the “PortView(s)” fields in both the request and response JSON reference in the following pages represent the bits associated with the PortViews that are being flip turned on (i.e. a 1) to signify selected. Since, we have 16 Port Views there are 16 bits representing each port view. The lowest most end bit-1 represents PortView1 and the highest most bit-16 represents PortView 16.

You can use bitwise “AND” and “OR” operators to determine which bits are selected and which to select. Below is the mapping between the Port Views and their associated bits.

Port Views associated bits

Port Views	Associated bit	Number Value
1	0000 0000 0000 0001	1
2	0000 0000 0000 0010	2
3	0000 0000 0000 0100	4
4	0000 0000 0000 1000	8
5	0000 0000 0001 0000	16
6	0000 0000 0010 0000	32
7	0000 0000 0100 0000	64
8	0000 0000 1000 0000	128
9	0000 0001 0000 0000	256
10	0000 0010 0000 0000	512
11	0000 0100 0000 0000	1024
12	0000 1000 0000 0000	2048
13	0001 0000 0000 0000	4096
14	0010 0000 0000 0000	8192
15	0100 0000 0000 0000	16384
16	1000 0000 0000 0000	32768

To select PortView1 and Portview3 you would specify a 3 (bitwise 1 | 2).

3.1 POST http://DP-Studio-IP:5000/api/v1/validation-object

JSON Request Body (`application/json`)

```
{
  "ValidationObjName": string // Validation Object Name created in the DP-Studio GUI
  “User”: string, // username of the tester role user can be emptied string i.e. “”
  "PortView":number // Number represents the bit mask associated with the port
}
```

JSON Response Body (`application/json`)

```
{
   "ConditionResult": true|false, // Condition check same as clicking on Validate button
   "ValidationImageTraceFailurePath": "\\\\StudioName\\ArchivedImages... ",
   "ValidationObjName": "Name", // Name of evaluated Validation Object
```

```
  "ValidationResult": "Failure|Success" // The result of Validation Object
}
```

3.2 POST http://DP-Studio-IP:5000/api/v1/ ui-set

JSON Request Body (`application/json`)

```
{
 "UISetName": string, //UISet Name as created in DP-Studio GUI
 “User”: string, // username of the tester role user can be emptied string i.e “”
 "PortViews": integer // integer represents the bit mask associated with the ports
}
```

JSON Response Body (`application/json`)

```
{
  "ResultCode": 0, // 0 for successfully executing the UISet
  "UISetName": "Name" // Name of executed UISet
}
```

3.3 POST http://DP-Studio-IP:5000/api/v1/ui-test-case

JSON Request Body (`application/json`)

```
{
 "TestCaseName": string // Name of Test Case as created in DP-Studio GUI
}
```

JSON Response Body (`application/json`)

```
{
  "ResultCode": integer, // 0 is Success for executing UITestCase non-zero error
  "TestCaseName": string // Name of executed Test Case
}
```

3.4 POST http://DP-Studio-IP:5000/api/v1/ui-job

JSON Request Body (`application/json`)

```
{
```

```
  "JobName": "Name" // Name of UI Job as created in DP-Studio GUI
}
```

JSON Response Body (`application/json`)

```
{
   "JobName": "Name", // Name of executed UIJob
   "ResultCode": 0 // 0 is Success for executingUI Job object
}
```

3.5 POST http://DP-Studio-IP:5000/api/v1/job-run

JSON Request Body (`application/json`)

```
{
  "JobRunName": string, // // Name of JobRun as created in DP-Studio GUI
  "PortViews":integer, // Integer represents the bit mask associated with the ports
  "StartTime": "YYYY-MM-DD HH:MM:SS", // Start time can be before time now
  "EndTime": "YYYY-MM-DD HH:MM:SS", // End time has to be after start time and current time
  "Duration":string, // None, Continuous, Hourly, Daily, Weekly and Monthly
  "StartMode":string, // parallel or series applicable to running JobRun on multi devices
  "DelayTime": integer // delay time milliseconds between JobRun running on multiple devices
}
```

JSON Response Body (`application/json`)

```
{
   "JobName": "JR.ClosedCaption",
   "ResultCode": 0
}
```

3.6 POST http://DP-Studio-IP:5000/api/v2/port-views

JSON Request Body (`application/json`)

```
{
  "Operation": integer, // 1 = Get Port Views assigned to user
                        // 2 = Lock/Unlock Port Views,
                        // 3 = Get Locked/Unlock Port Views

  “User”:string, //User associated with the above “Operation” i.e. 1, 2 and 3

  //LockPortViews to Lock/Unlock applicable only when operation = 2
  "LockPortViews":integer, // integer (bit-mask) indicating selected ports for operation 2
  "LockMode":integer, // 0=Unlock, 1=Lock applicable to when operation = 1, 2 o 3
```

}

Response Object
{
"PortViews": integer, //integer (bit-mask) indicating ports that were successfully locked or unlocked based on Lock Operation as well as port views assigned to user if operation is get Port View Mapping
"ResultCode": integer // 0 for success non zero error code calling the API
}

JSON Response Body (`application/json`)
{
"PortViews": integer, //integer (bit-mask) indicating ports that were successfully locked or unlocked based on Lock Operation as well as port views assigned to user if operation is get Port View Mapping
"ResultCode": 0 // 0 for success non zero error code calling the API
}

*Note in the above example “LockPortViews” and “LockMode” fields are not being used since operation is a 1. However, it needs to there as a placeholder for now and will be used when the operation is a 2 or a 3. In this example the REST API call will return all the Port Views assign to the tester9 user. The “PortViews” field in the response indicate a value of 4121 which translate to the following bits “0001000000011001” which means Port Views 1, 4, 5, and 13 are assigned to this user.

3.7 POST http://DP-Studio-IP:5000/api/v3/text

JSON Request Body (`application/json`)

```
{
 "PortView":integer, //bit-mask integer identify the port-view
 "Operation": integer,//1=Search, 2=Extract
 "SearchText": string, //search string use when operation=1
 "xLeftTopRectangle":integer, //X - Left top corner
 "yLeftTopRectangle":integer,//Y – Left top corner
 "RectangleWidth":integer,//width of rectangle
 "RectangleHeight"integer, //height of rectangle
 "Threshold": integer//percent integer (0 to 100) indicate minimum match or extraction threshold
}
```

JSON Response Body (`application/json`)

```
{
"ResultCode": integer, //0 for success.  Anything other 0 might indicate issues
"ResultCodeMsg": string,//Error message if ResultCode is non zero
"ResultFlag": boolean,//true for found text or successful text extraction
"ResultScore":number,//percent indicating what the actual match or extraction score
"ResultText": "No Signal" //actual match or extracted text from the screen
}
```

3.8 POST http://DP-Studio-IP:5000/api/v3/image

JSON Request Body (`application/json`)

```
{
  "PortView": integer, //bit-mask integer represent selected Port View
  "SearchImage": string, //UNC file path of search image file
  "xLeftTopRectangle": integer, //X left top corner
  "yLeftTopRectangle": integer, //Y left top corner
  "RectangleWidth": integer, // width of rectangular search region
  "RectangleHeight": integer, //height of rectangular search region
  "Threshold": number //float 0 to 1 indicates matching threshold where 1 is perfect match
}
```

JSON Response Body (`application/json`)

```
{
   "ResultCode": integer, // any value non-zero is not a success
   "ResultCodeMsg": "", //Message if result code is non-zero
   "ResultFlag": boolean, // true if image match false if it does not
   "ResultScore": number //float from 0 to 1 indicates actual matching score
}
```

3.9 POST http://DP-Studio-IP:5000/api/v4/video

The video REST API exposes the features of the Smart-Validation objects of ValidateVideo (freeze and pixelation detection) and ValidateAudio (Audio presence detection) as well as video recording, image snapshot and video source information into one REST API. All the fields in the Request JSON body below **need to be present for all operations** even though not all fields will be used by each operation. For fields that are not used by a given operation you can use “” for

string fields and 0 for number fields. There are six operations (listed below) a tester/developer could perform.

1. Operation-1: Get information about the source video. Whether video source is connected, video resolution and frame rate etc.
2. Operation-2: Record a video clip of source video.
3. Operation-3: Take snapshots of regions of the input video. Snapshots can be fed into image REST API for image search or for image trace archiving purposes.
4. Operation-4: Detect video freeze.
5. Operation-5: Detect video pixelation. This operation is CPU intensive and should be used with caution.
6. Operation-6: Detect audio presence.

JSON Request Body (`application/json`)

```
{
 "PortView": integer,    // bit-mask integer identify the port-view
 "Operation": integer,  // 1 = Get Video Info, 2 = Record Video Clip, 3 = Get Video Snapshot,
                        // 4 = Freeze Detection, 5 = Pixelation Detection, 6 = Audio Cutoff

 //Operation = 2
 "RecordDuration":integer, //Record duration in milliseconds

 //Operation 2 and 3
 "OutputFilePath”:string, //Specify the fully qualified UNC path to the saved MP4 or PNG file
                          //Make sure the DP-Studio can access the UNC path.

 //Operation 3, 4, and 5
 //Use Video Zoom Window and “Design” mode to rubber band and define the regions for
 //snapshot or regions to monitor
 "xLeftTopRectangle":integer, X Left Top Rectangle
 "yLeftTopRectangle":integer, Y Left Top Rectangle
 "RectangleWidth":integer, width of rectangle region (1 to 960)
 "RectangleHeight":integer, height of rectangle region (1 to 540)

 //Operation 4 and 5
 "Threshold":float, //0 to 1 for operation 4 and 5. See Note1 below.

 //Applies to operation 4 and 6
 "Duration":integer, // milliseconds to test for event applies to operation 4 and 6 only. See Note2
 "MonitorDuration":integer, //milliseconds period to monitor. See Note3
 // Operation 6
```

```
  "AudioCutOffThreshold":integer // negative integers for dB. Use -50 dB cutoff. See Note4
}
```

Note1:
"Threshold" field. For freeze detection recommended value is 0.99 (remember to have 0 in front of decimal to be treated as number). For Pixelation recommended value is 15. Where 15 is number of pixelated regions.

Note2:
"Duration" field applies to operation 4 and 6 only. In operation 4 it is the duration of continuous video freeze and in operation 6 it is the duration of no audio. Recommended value is 10 to 15 seconds.

Note3:
The "MonitorDuration" field needs to be larger than the "Duration" field. Recommended value is 30 to 60 seconds.

Note4:
"AudioCutOffThreshold" field represents a negative decibel value integer. Recommend value for audio cutoff is -50 dB.

JSON Responses below by Operation

Operation = 1
JSON Response Body (`application/json`)

```
{
   "Connected": boolean, // true indicates connected signal false no signal detected
   "FrameRate": float, // frame rate in fps
   "Height": integer, // Video source Height resolution
   "Width": integer // Video Source Width resolution
   "ResultCode": integer, // 0 = Success Non 0 indicates error
   "VideoFormat":string, // Video Format; SD, HD and Full HD
}
```

Operation = 2
JSON Response Body (`application/json`)

```
{
   "ResultCode": integer, // 0 = Success Non 0 indicates error
   "ResultFilePath":string // UNC path to  MP4 recorded file.  Make sure UNC path supplied in
                           // JSON Request is accessible by the requested DP-Studio
}
```

Operation = 3
JSON Response Body (`application/json`)

```
{
   "ResultCode": integer, // 0 = Success Non 0 indicates error
   "ResultFilePath":string, // UNC path to  PNG snapshot file.  Make sure UNC path supplied in
                                 // JSON Request is accessible by the requested DP-Studio
   "ResultFlag": boolean,  // True snapshot was successful False snapshot was not successful
}
```

Operation = 4
JSON Response Body (`application/json`)

```
{
   "ResultCode": integer, // 0 = Success Non 0 indicates error
   "ResultFlag": false, // True = Freeze, False = No Freeze
}
```

Operation = 5
JSON Response Body (`application/json`)

```
{
   "ResultCode": integer, // 0 = Success Non 0 indicates error
   "ResultFlag": false, // True = Pixelation, False = No Pixelation
}
```

Operation = 6
JSON Response Body (`application/json`)

```
{
   "ResultCode": 0, // 0 = Success Non 0 indicates error
   "ResultFlag": false, // True Audio presence false = No Audio presence
}
```